WIDROW-HOFF LMS ADALINE DEMONSTRATOR FOR SCHOOLS AND COLLEGES

Stephen R. Alty and Clive Cheong Took

Department of Electronic Engineering,
Royal Holloway, University of London,
Egham TW20 0EX, UK.

ABSTRACT

The Widrow-Hoff LMS (or 'Adaline') algorithm developed originally in 1960 is fundamental to the operation of countless signal processing machine learning systems in use even today. Bernard Widrow and Ted Hoff famously developed an Adaline machine demonstrator using basic analog off the shelf components to show how a 'perceptron' could be trained manually [1]. This paper details the design and development of a fully digital Adaline Least-Mean-Square algorithm demonstrator. The simplistic design presented here is completely open-source with all code, bill of materials and 3D casing models available at this Github Repository for download and reproduction. The demonstrator enables quick visualisation of the training and testing of a simple perceptron algorithm running on the inexpensive Arduino platform. The total costs of the device is estimated to be less than $60 and could be used in classrooms and colleges the world-over to demonstrate the seminal work of Widrow and Hoff [2] to a wide audience.

Index Terms— Widrow-Hoff, Adaline, LMS, Educational Demonstrator

1. INTRODUCTION

Bernard Widrow and Ted Hoff are famous in the world of the signal processing when they published their seminal work on the Adaline or Least-Means-Squared (LMS) algorithm that changed the world of DSP and Machine Learning overnight. The same algorithms (and their multitude of derivatives) are used today in many different applications and have become entirely *ubiquitous*. Yet, it can be hard for the novice to visualize and fully comprehend how the algorithm actually works. Firstly, the equations may appear quite complex to someone who is uninitiated in the realms of partial differential mathematics. The aim of this work is to break down the barriers to understanding and visualizing how the algorithm works in much the same way that Widrow and Hoff sought to achieve in the analog Adaline demonstrator and subsequent variant based on 'mem-istors'. This follows our pedagogical approach to creativity for student learning [3].

To the best of the authors' knowledge there is no *educational* and *reproducible* version of an 'Adaline' demonstrator device shown in Figure 1, which was originally conceived and built by Widrow and Hoff [2]. Other than relying on the Widrow-Hoff LMS algorithm, the design presented here is wholly new. The novelty is in the value of the design and that it is available freely for educational purposes. On the other hand, related hardware implementations of Adaline were mostly aimed for research (rather than educational) purposes such as in FPGA implementation [4], and in the use of FFT for fast computation in Adaline [5].

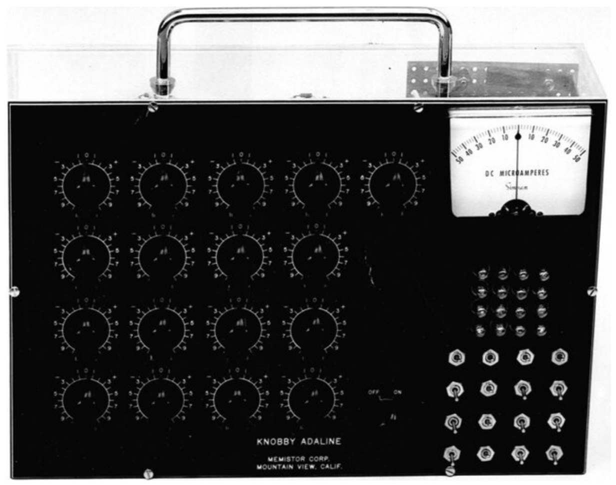

Fig. 1. Image of the 'Knobby'-Adaline device built by Ted Hoff in 1960 [6].

We have developed a simple design based on *off-the-shelf* components that are inexpensive and easy to construct. The 'Arduino Uno' micro-controller forms the brains of the system, storing the input data, the weight vector and runs the LMS algorithm to update the weight vector on the push of a button. The neuron output is displayed on an RGB LCD display that changes color from blue to red to show binary classes. There are two training buttons, a blue one for '+1' target class and a red one for '–1' target class. The total cost

Thanks to David Chapman for help with the 3D casing design.
Corresponding author `steve.alty@rhul.ac.uk`

of the system is estimated to be around \$60 and the design files are available from this Github Repository for download. Our design is fully open-source to enable others to reproduce the device to help disseminate the Widrow-Hoff concept in an easy to use demonstrator. The hope is that schools and colleges may make their own versions and possibly improve and develop the design in the future.

2. BACKGROUND

The Widrow-Hoff algorithm is well-known in the circles of signal processing and machine learning but it is included here for completeness. Essentially, at the heart of the system is an adaptive linear element (hence the term 'Adaline' [7]) consisting of an adaptive linear combiner, sometimes with a hard quantizer which produces a binary output of ± 1 which can be considered to be a 'class'. The object being to recognise the

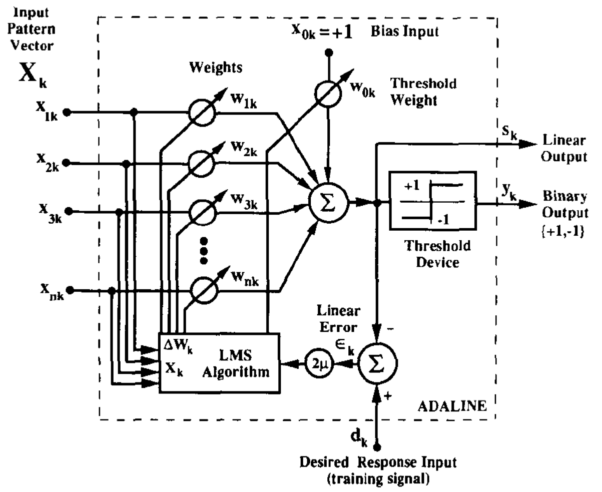

Fig. 2. The LMS adaptation process in Adaline [7].

k^{th} input pattern, $\mathbf{x}_k$, by adapting a vector of weights, $\mathbf{w}_k$, automatically to group different input patterns in respective classes. In this case we focus on a binary classification process (i.e. two groups of patterns). This system is depicted in Figure 2 and can also be defined mathematically as the inner product of the two vectors plus an offset or bias element w_0:

$$s_k = \mathbf{w}_k^T \mathbf{x}_k + w_{0k} \tag{1}$$

$$y_k = sgn(s_k) \tag{2}$$

where the weight vector, $\mathbf{w}_k = [w_1, w_2, w_3, \ldots, w_n]^T$, the input vector, $\mathbf{x}_k = [x_1, x_2, x_3, \ldots, x_n]^T$, and the signum $sgn(\cdot)$ function performs the binary classification, y_k. The output of the linear summing element, s_k, is then compared to that of the desired output, d_k, (to effect learning) and an error signal is formed,

$$e_k = d_k - s_k = d_k - \mathbf{w}_k^T \mathbf{x}_k \tag{3}$$

We then wish to minimise the *squared* error, as this creates a convex error curve enabling the gradient descent procedure to function correctly, thus,

$$e_k^2 = d_k^2 + \mathbf{w}_k^T \mathbf{x}_k \mathbf{x}_k^T \mathbf{w}_k - 2 d_k \mathbf{w}_k^T \mathbf{x}_k \tag{4}$$

Next we take the *partial differential* of the squared error with respect to the weight vector as it is this value that we wish to optimize

$$\begin{aligned} \frac{\partial e_k^2}{\partial \mathbf{w}_k} &= 2\mathbf{x}_k \mathbf{x}_k^T \mathbf{w}_k - 2 d_k \mathbf{x}_k \\ &= 2(\mathbf{x}_k^T \mathbf{w}_k - d_k)\mathbf{x}_k \\ &= -2 e_k \mathbf{x}_k \end{aligned} \tag{5}$$

Hence, using this result yields the update rule as follows:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu e_k \mathbf{x}_k \tag{6}$$

where $\mathbf{x}_k$ indicates the k^{th} input pattern vector, $\mathbf{w}_k$ the k^{th} weight vector value and $\mathbf{w}_{k+1}$ represents the future iteration of the weight vector after one training cycle, and μ is the adaptation coefficient. The training process typically needs to be run over a number of iterations depending on the patterns, the length of the weight vector and the adaption coefficient μ. Selecting a small value for μ will mean adaption proceeds slowly and many update cycles will be required. Selecting a large value for μ, however, can lead to instability and the weight vector may not converge. The appropriate selection of the value for μ is outside the realms of this paper, suffice to say that trial and error is often used to select a suitable value for a given system.

3. THE DIGITAL ADALINE DEMONSTRATOR

The mathematics behind the LMS update rule is very simple and thus lends itself to real-time implementation. Here we have used the Arduino Uno micro-controller platform to interface the peripherals but also run the LMS algorithm at intervals when the appropriate training button is pressed.

Referring to Figure 3, a pattern can be entered on the front panel 4x4 matrix of buttons. Pressing a button will toggle its color from red to blue and another press will revert back to red. Red buttons correspond to a -1 being entered in the pattern vector $\mathbf{x}_k$ and blue buttons correspond to a $+1$. The RGB liquid crystal display (LCD) shows the output of the Adaline computation (either positive or negative) and the backlight is programmed to change from red to blue to match a -1 or $+1$ output 'class' respectively. The output value displayed on the LCD has been converted to a sigmoid output using the hyperbolic tangent function. This also gives a sense for the *strength* of classification – or more accurately – the distance of the input pattern from the hyperplane represented by the weight vector:

$$output = \tanh(s_k) \tag{7}$$

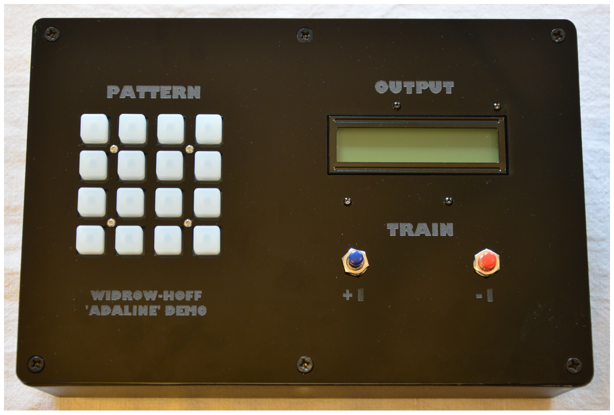

Fig. 3. 'Digital' Adaline device built at Royal Holloway, University of London, [Github Repository].

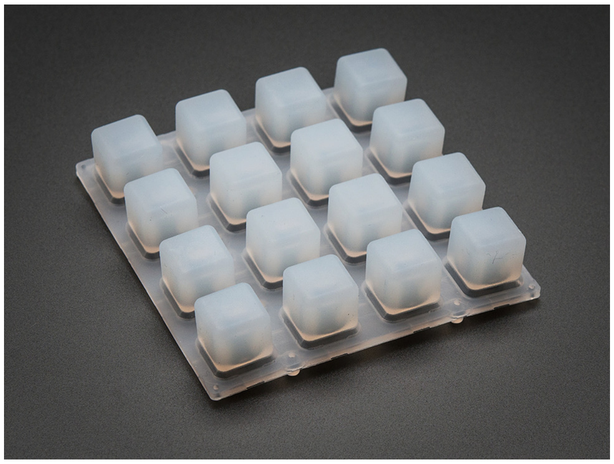

Fig. 4. 'NeoTrellis' 4x4 keypad with embedded NeoPixels with silicone rubber cover available here and here.

The device is made from off-the-shelf components to make building a copy of this device as easy as possible. The keypad is sourced from AdaFruit Industries at a cost of approx $12.50 along with the silicone elastomer and LED diffuser cover see Figure 4 costing around $5. Together these items form the input vector for the digital Adaline and are interfaced using the I2C protocol to the Arduino via a Grove I2C Base Shield, see Figure 5 at a cost of approx. $3.50. An RGB LCD at a cost of $15 (see Figure 6) is also connected to the Grove I2C Shield to show the output and change color according to the class of the input pattern. Using the I2C interface makes the assembly of the device essentially plug and play. Finally, two simple momentary push buttons – one blue and one red – to enable the training process, are connected to two digital inputs on the Arduino Uno completing the physical build of the device. The whole unit was housed in a simple ABS plastic case and a front panel was designed to accommodate the 4x4 silicone elastomer keypad, the RGB LCD and the two push

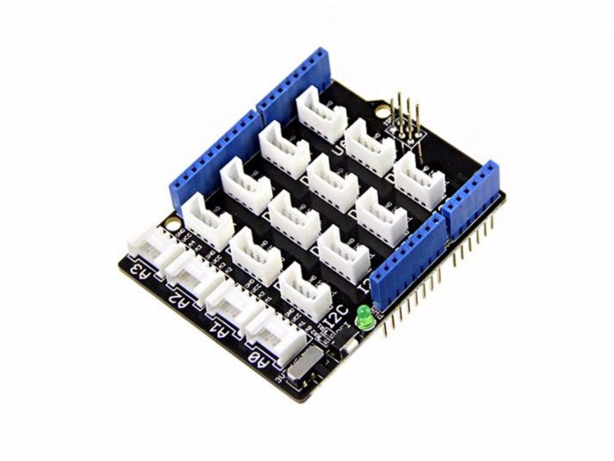

Fig. 5. Seeed Studio's Grove Base Shield V2 available here.

Fig. 6. Seeed Studios RGB LCD 16x2 Character Display available here.

Item	**Cost**
Arduino Uno (Clone)	$17.00
Seeed Grove Shield V2	$3.50
Seeed RGB LCD	$15.00
AdaFruit NeoTrellis 4x4	$12.50
AdaFruit Silicone Keypad	$5.00
Momentary SPST button x2	$1.00
ABS Case 220x150x64mm	$8.00
Total	$62.00

Table 1. Bill of materials with approximate costings

buttons. Since the top panel was laser cut – it was also etched to show 'Pattern', 'Output' and 'Train' buttons along with a title, 'Widrow-Hoff 'Adaline' Demo'. The 'DXF' file for the top panel is also included in the repository.

The internal coding of the digital Adaline demonstrator was programmed using the Arduino IDE. The code (though not included here for brevity) is fully commented and freely available along with all the information required to source and assemble a complete Adaline demonstrator in here.

4. TRAINING PROCESS

When the Adaline device is first turned on, each keypad illuminates red to indicate that the pattern vector is simply a series of '-1s', this is effectively a blank pattern. The weight vector is initialised with pseudo-random numbers between ± 1, this is important to enable consistent adaptation from the start. The user can then input a specific pattern on the keypad by toggling the buttons to blue (this will change the input vector element to '$+1$'). Widrow demonstrated his 'Knobby' and 'Mem-istor' Adaline devices usually by training them to recognise 'J' and 'T'-type letter shaped patterns. We can do the same here and enter a blue shaped 'J' pattern and then press the -1 'train' button to train the output to belong to the negative class for this pattern. Then, we can change the pattern to a 'T' and press the $+1$ 'train' button to train the output to belong to the positive class for this pattern. The training sub-routine has a deliberate delay introduced to slow the update process so the adaption process is visible to the operator. Widrow went on to enter variations of the original patterns (typically a translation of the 'J' and 'T' by one 'pixel' or sometimes a rotated 'J' or 'T') and trained these to match the output classes accordingly.

Pressing the train buttons runs the LMS algorithm which steadily updates the internal weight vector using the rule in Eq. (6). More specifically, pressing the $+1$ button presents a positive desired output class, i.e. $d_k = +1$ in Eq. (3) and conversely pressing the -1 button presents a negative desired output class, i.e. $d_k = -1$. In this way, it is easy to train the device to recognize many different patterns across the 4x4 colored keypad array. The demonstration develops the following learning outcomes; 1) To be able to implement a perceptron in hardware; 2) To understand the role of the nonlinear activation in decision making of Adaline; 3) To evaluate the quality of training data; 4) To apply Adaline to a real-world problem. A full demonstration of the training process is available in this repository.

5. CONCLUSION

A complete digital recreation of Widrow and Hoff's famous Adaline demonstrator device has been developed. The intention is to enable others – perhaps less technically able – to acquire the components and assemble their own devices in an attempt to introduce a wider audience to the world of adaptive signal processing and the principles of machine learning. To this end, its *reproducible* materials are made available in this repository, whilst its design takes into account the *accessibility* and *affordability* of the hardware required to construct a digital version of the Widrow-Hoff Adaline.

6. REFERENCES

[1] Bernard Widrow, "An adaptive "Adaline" neuron using chemical memistors," Tech. Rep., Stanford Electronic Labs, Stanford, USA, 1960.

[2] Bernard Widrow and Marcian (Ted) Hoff, "Adaptive switching circuits," in *IRE Western Electric Show and Convention Record, Part 4, pages 96-104*, 1960.

[3] Clive Cheong Took, Stephen R. Alty, Anush Yardim, and David M. Howard, "Creativity first, Science follows: Lessons in Digital Signal Processing education," *IEEE Signal Processing Magazine*, vol. 38, no. 3, pp. 51–61, 2021.

[4] Omid Sharifi-Tehrani and Mohsen Ashourian, "An FPGA-based implementation of Adaline neural network with low resource utilization and fast convergence," *PRZEGLAD ELEKTROTECHNICZNY*, vol. 86, no. 12, pp. 288–292, 2010.

[5] Zai Peng Goh, Mohd Amran Mohd Radzi, Yee Von Thien, Hashim Hizam, and Noor Izzri Abdul Wahab, "Hybrid FFT-Adaline algorithm with fast estimation of harmonics in power system," *Signal Processing*, vol. 10, no. 8, pp. 855–864, 2016.

[6] Alexander Magoun, "A nonrandom walk down memory lane with Bernard Widrow [scanning our past]," in *Proceedings of the IEEE*. IEEE, 2014, vol. 102, pp. 1622–1629.

[7] Bernard Widrow and Michael A. Lehr, "30 years of adaptive neural networks: Perceptron, Madaline, and Backpropagation," in *Proceedings Title*. IEEE, 1990, vol. 78, pp. 1415–1442.

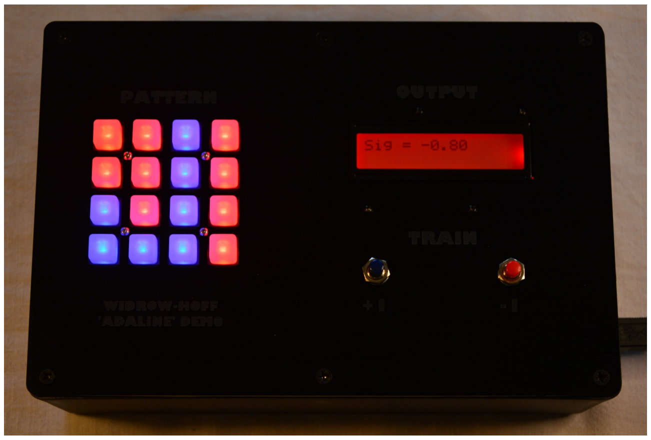

Fig. 7. Training the 'J' pattern to give negative output.

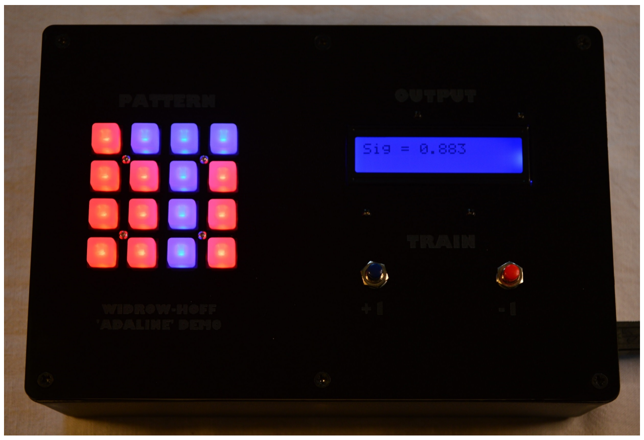

Fig. 8. Training the 'T' pattern to give positive output.